UNITED STATES PATENT APPLICATION

For

TAG DATA STRUCTURE FOR MAINTAINING RELATIONAL DATA OVER CAPTURED OBJECTS

Inventors:

Erik de la Iglesia
Rick Lowe
Ratinder Paul Singh Ahuja
Shaun Coleman
Samuel King
Ashish Khasgiwala

Prepared by:
Blakely, Sokoloff, Taylor & Zafman
12400 Wilshire Boulevard
Seventh Floor
Los Angeles, California 90025
(408) 947-8200

Attorney's Docket No. 6897P003

"Express Mail" mailing label number: EV410138225US
Date of Deposit: March 30, 2004
I hereby certify that I am causing this paper or fee to be deposited with the United States Postal Service "Express Mail Post Office to Addressee" service on the date indicated above and that this paper or fee has been addressed to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

Vineta T. Tufono
(Typed or printed name of person mailing paper or fee)

(Signature of person mailing paper or fee) 3-30-2004

(Date signed)

TAG DATA STRUCTURE FOR MAINTAINING RELATIONAL DATA OVER CAPTURED OBJECTS

PRIORITY AND RELATED APPLICATIONS

[0001] This patent application is related to, incorporates by reference, and claims the priority benefit of U.S. Provisional Application 60/528,680, entitled "TAG DATA STRUCTURE FOR MAINTAINING RELATIONAL DATA OVER CAPTURED APPLICATION DATA", filed December 10, 2003, and of U.S. Provisional Application 60/528,644, entitled "VERIFYING CAPTURED OBJECTS BEFORE PRESENTATION", filed December 10, 2003.

FIELD OF THE INVENTION

[0002] The present invention relates to computer networks, and in particular, to capturing and indexing objects in a computer network.

BACKGROUND

[0003] Computer networks and systems have become indispensable tools for modern business. Modern enterprises use such networks for communications and for storage. The information and data stored on the network of a business enterprise is often a highly valuable asset. Modern enterprises use numerous tools to keep outsiders, intruders, and unauthorized personnel from accessing valuable information stored on the network. These tools include firewalls, intrusion detection systems, and packet sniffer devices. However, once an intruder has gained access to sensitive content, there is no

network device that can prevent the electronic transmission of the content from the network to outside the network. Similarly, there is no network device that can analyse the data leaving the network to monitor for policy violations, and make it possible to track down information leeks. What is needed is a comprehensive system to capture, store, and analyse all data communicated using the enterprises network.

SUMMARY OF THE INVENTION

[0004] Content leaving a local network can be captured. Objects captured over a network by a capture system can be indexed to provide enhanced search and content analysis capabilities. In one embodiment the objects can be indexed using a data structure having a source address field to indicate an origination address of the object, a destination address field to indicate a destination address of the object, a source port field to indicate an origination port of the object, a destination port field to indicate a destination port of the object, a content field to indicate a content type from a plurality of content types identifying a type of content contained in the object, and a time field to indicate when the object was captured. The data structure may also store a cryptographic signature of the object to ensure the object is not altered after capture.

BRIEF DESCRIPTION OF THE DRAWINGS

[0005] The present invention is illustrated by way of example, and not by way of limitation, in the figures of the accompanying drawings in which like reference numerals refer to similar elements and in which:

[0006] **Figure 1** is a block diagram illustrating a computer network connected to the Internet;

[0007] **Figure 2** is a block diagram illustrating one configuration of a capture system according to one embodiment of the present invention;

[0008] **Figure 3** is a block diagram illustrating the capture system according to one embodiment of the present invention;

[0009] **Figure 4** is a block diagram illustrating an object assembly module according to one embodiment of the present invention;

[0010] **Figure 5** is a block diagram illustrating an object store module according to one embodiment of the present invention;

[0011] **Figure 6** is a block diagram illustrating an example hardware architecture for a capture system according to one embodiment of the present invention; and

[0012] **Figure 7** is a flow diagram illustrating object verification according to one embodiment of the present invention.

DETAILED DESCRIPTION

[0013] Although the present system will be discussed with reference to various illustrated examples, these examples should not be read to limit the broader spirit and scope of the present invention. Some portions of the detailed description that follows are presented in terms of algorithms and symbolic representations of operations on data within a computer memory. These algorithmic descriptions and representations are the means used by those skilled in the computer science arts to most effectively convey the substance of their work to others skilled in the art. An algorithm is here, and generally, conceived to be a self-consistent sequence of steps leading to a desired result. The steps are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared and otherwise manipulated.

[0014] It has proven convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers or the like. It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise, it will be appreciated that throughout the description of the present invention, use of terms such as "processing", "computing", "calculating", "determining", "displaying" or the like, refer to the action and processes of a computer system, or similar electronic computing device, that manipulates and transforms data represented as physical (electronic) quantities within the computer system's registers and memories into other data similarly represented

as physical quantities within the computer system memories or registers or other such information storage, transmission or display devices.

[0015] As indicated above, one embodiment of the present invention is instantiated in computer software, that is, computer readable instructions, which, when executed by one or more computer processors/systems, instruct the processors/systems to perform the designated actions. Such computer software may be resident in one or more computer readable media, such as hard drives, CD-ROMs, DVD-ROMs, read-only memory, read-write memory and so on. Such software may be distributed on one or more of these media, or may be made available for download across one or more computer networks (e.g., the Internet). Regardless of the format, the computer programming, rendering and processing techniques discussed herein are simply examples of the types of programming, rendering and processing techniques that may be used to implement aspects of the present invention. These examples should in no way limit the present invention, which is best understood with reference to the claims that follow this description.

Networks

[0016] Figure 1 illustrates a simple prior art configuration of a local area network (LAN) 10 connected to the Internet 12. Connected to the LAN 102 are various components, such as servers 14, clients 16, and switch 18. There are numerous other known networking components and computing devices that can be connected to the LAN 10. The LAN 10 can be implemented using various wireline or wireless technologies, such as Ethernet and 802.11b. The LAN 10 may be much more complex than the simplified diagram in Figure 1, and may be connected to other LANs as well.

[0017] In Figure 1, the LAN 10 is connected to the Internet 12 via a router 20. This router 20 can be used to implement a firewall, which are widely used to give users of the LAN 10 secure access to the Internet 12 as well as to separate a company's public Web server (can be one of the servers 14) from its internal network, i.e., LAN 10. In one embodiment, any data leaving the LAN 10 towards the Internet 12 must pass through the router 12. However, there the router 20 merely forwards packets to the Internet 12. The router 20 cannot capture, analyse, and searchably store the content contained in the forwarded packets.

[0018] One embodiment of the present invention is now illustrated with reference to Figure 2. Figure 2 shows the same simplified configuration of connecting the LAN 10 to the Internet 12 via the router 20. However, in Figure 2, the router 20 is also connected to a capture system 22. In one embodiment, the router 12 splits the outgoing data stream, and forwards one copy to the Internet 12 and the other copy to the capture system 22.

[0019] There are various other possible configurations. For example, the router 12 can also forward a copy of all incoming data to the capture system 22 as well. Furthermore, the capture system 22 can be configured sequentially in front of, or behind the router 20, however this makes the capture system 22 a critical component in connecting to the Internet 12. In systems where a router 12 is not used at all, the capture system can be interposed directly between the LAN 10 and the Internet 12. In one embodiment, the capture system 22 has a user interface accessible from a LAN-attached device, such as a client 16.

[0020] In one embodiment, the capture system 22 intercepts all data leaving the network. In other embodiments, the capture system can also intercept all data being

communicated inside the network 10. In one embodiment, the capture system 22 reconstructs the documents leaving the network 10, and stores them in a searchable fashion. The capture system 22 can then be used to search and sort through all documents that have left the network 10. There are many reasons such documents may be of interest, including network security reasons, intellectual property concerns, corporate governance regulations, and other corporate policy concerns.

Capture System

[0021] One embodiment of the present invention is now described with reference to Figure 3. Figure 3 shows one embodiment of the capture system 22 in more detail. The capture system 22 is also sometimes referred to as a content analyzer, content or data analysis system, and other similar names. In one embodiment, the capture system 22 includes a network interface module 24 to receive the data from the network 10 or the router 20. In one embodiment, the network interface module 24 is implemented using one or more network interface cards (NIC), e.g., Ethernet cards. In one embodiment, the router 20 delivers all data leaving the network to the network interface module 24.

[0022] The captured raw data is then passed to a packet capture module 26. In one embodiment, the packet capture module 26 extracts data packets from the data stream received from the network interface module 24. In one embodiment, the packet capture module 26 reconstructs Ethernet packets from multiple sources to multiple destinations for the raw data stream.

[0023] In one embodiment, the packets are then provided the object assembly module 28. The object assembly module 28 reconstructs the objects being transmitted by the packets. For example, when a document is transmitted, e.g. as an email attachment, it

is broken down into packets according to various data transfer protocols such as Transmission Control Protocol/Internet Protocol (TCP/IP) and Ethernet. The object assembly module 28 can reconstruct the document from the captured packets.

[0024] One embodiment of the object assembly module 28 is now described in more detail with reference to Figure 4. When packets first enter the object assembly module, they are first provided to a reassembler 36. In one embodiment, the reassembler 36 groups – assembles – the packets into unique flows. For example, a flow can be defined as packets with identical Source IP and Destination IP addresses as well as identical TCP Source and Destination Ports. That is, the reassembler 36 can organize a packet stream by sender and recipient.

[0025] In one embodiment, the reassembler 36 begins a new flow upon the observation of a starting packet defined by the data transfer protocol. For a TCP/IP embodiment, the starting packet is generally referred to as the "SYN" packet. The flow can terminate upon observation of a finishing packet, e.g., a "Reset" or "FIN" packet in TCP/IP. If now finishing packet is observed by the reassembler 36 within some time constraint, it can terminate the flow via a timeout mechanism.. In an embodiment using the TPC protocol, a TCP flow contains an ordered sequence of packets that can be assembled into a contiguous data stream by the ressembler 36. Thus, in one embodiment, a flow is an ordered data stream of a single communication between a source and a destination.

[0026] The flown assembled by the reassember 36 can then be provided to a protocol demultiplexer (demux) 38. In one embodiment, the protocol demux 38 sorts assembled flows using the TCP Ports. This can include performing a speculative

classification of the flow contents based on the association of well-known port numbers with specified protocols. For example, Web Hyper Text Transfer Protocol (HTTP) packets – i.e., Web traffic – are typically associated with port 80, File Transfer Protocol (FTP) packets with port 20, Kerberos authentication packets with port 88, and so on. Thus in one embodiment, the protocol demux 38 separates all the different protocols in one flow.

[0027] In one embodiment, a protocol classifier 40 also sorts the flows in addition to the protocol demux 38. In one embodiment, the protocol classifier 40 – operating either in parallel or in sequence with the protocol demux 38 – applies signature filters to the flows to attempt to identify the protocol based solely on the transported data. Furthermore, the protocol demux 38 can make a classification decision based on port number which is subsequently overridden by protocol classifier 40. For example, if an individual or program attempted to masquerade an illicit communication (such as file sharing) using an apparently benign port such as port 80 (commonly used for HTTP Web browsing), the protocol classifier 40 would use protocol signatures, i.e., the characteristic data sequences of defined protocols, to verify the speculative classification performed by protocol demux 38.

[0028] In one embodiment, the object assembly module 28 outputs each flow organized by protocol, which represent the underlying objects. Referring again to Figure 3, these objects can then be handed over to the object classification module 30 (sometimes also referred to as the "content classifier") for classification based on content. A classified flow may still contain multiple content objects depending on the protocol used. For example, protocols such as HTTP (Internet Web Surfing) may contain over

100 objects of any number of content types in a single flow. To deconstruct the flow, each object contained in the flow is individually extracted, and decoded, if necessary, by the object classification module 30.

[0029] The object classification module 30 uses the inherent properties and signatures of various documents to determine the content type of each object. For example, a Word document has a signature that is distinct from a PowerPoint document, or an Email document. The object classification module 30 can extract out each individual object and sort them out by such content types. Such classification renders the present invention immune from cases where a malicious user has altered a file extension or other property in an attempt to avoid detection of illicit activity.

[0030] In one embodiment, the object classification module 30 determines whether each object should be stored or discarded. In one embodiment, this determination is based on a various capture rules. For example, a capture rule can indicate that Web Traffic should be discarded. Another capture rule can indicate that all PowerPoint documents should be stored, except for ones originating from the CEO's IP address. Such capture rules can be implemented as regular expressions, or by other similar means.

[0031] In one embodiment, the capture rules are authored by users of the capture system 22. The capture system 22 is made accessible to any network-connected machine through the network interface module 24 and user interface 34. In one embodiment, the user interface 34 is a graphical user interface providing the user with friendly access to the various features of the capture system 22. For example, the user interface 34 can provide a capture rule authoring tool that allows users to write and implement any capture

rule desired, which are then applied by the object classification module 30 when determining whether each object should be stored. The user interface 34 can also provide pre-configured capture rules that the user can select from along with an explanation of the operation of such standard included capture rules. In one embodiment, the default capture rule implemented by the object classification module 30 captures all objects leaving the network 10.

[0032] If the capture of an object is mandated by the capture rules, the object classification module 30 can also determine where in the object store module 32 the captured object should be stored. With reference to Figure 5, in one embodiment, the objects are stored in a content store 44 memory block. Within the content store 44 are files 46 divided up by content type. Thus, for example, if the object classification module determines that an object is a Word document that should be stored, it can store it in the file 46 reserved for Word documents. In one embodiment, the object store module 32 is integrally included in the capture system 22. In other embodiments, the object store module can be external – entirely or in part – using, for example, some network storage technique such as network attached storage (NAS) and storage area network (SAN).

Tag Data Structure

[0033] In one embodiment, the content store is a canonical storage location, simply a place to deposit the captured objects. The indexing of the objects stored in the content store 44 is accomplished using a tag database 42. In one embodiment, the tag database 42 is a database data structure in which each record is a "tag" that indexes an object in the content store 44 and contains relevant information about the stored object.

An example of a tag record in the tag database 42 that indexes an object stored in the content store 44 is set forth in Table 1:

Table 1

Field Name	Definition
MAC Address	Ethernet controller MAC address unique to each capture system
Source IP	Source Ethernet IP Address of object
Destination IP	Destination Ethernet IP Address of object
Source Port	Source TCP/IP Port number of object
Destination Port	Destination TCP/IP Port number of the object
Protocol	IP Protocol that carried the object
Instance	Canonical count identifying object within a protocol capable of carrying multiple data within a single TCP/IP connection
Content	Content type of the object
Encoding	Encoding used by the protocol carrying object
Size	Size of object
Timestamp	Time that the object was captured
Owner	User requesting the capture of object (rule author)
Configuration	Capture rule directing the capture of object
Signature	Hash signature of object
Tag Signature	Hash signature of all preceding tag fields

[0034] There are various other possible tag fields, and some embodiments can omit numerous tag fields listed in Table 1. In other embodiments, the tag database 42 need not be implemented as a database, and a tag need not be a record. Any data structure capable of indexing an object by storing relational data over the object can be used as a tag data structure. Furthermore, the word "tag" is merely descriptive, other names such as "index" or "relational data store," would be equally descriptive, as would any other designation performing similar functionality.

[0035] The mapping of tags to objects can, in one embodiment, be obtained by using unique combinations of tag fields to construct an object's name. For example, one such possible combination is an ordered list of the Source IP, Destination IP, Source Port,

Destination Port, Instance and Timestamp. Many other such combinations including both shorter and longer names are possible. In another embodiment, the tag can contain a pointer to the storage location where the indexed object is stored.

[0036] The tag fields shown in Table 1 can be expressed more generally, to emphasize the underlying information indicated by the tag fields in various embodiments. Some of these possible generic tag fields are set forth in Table 2:

Table 2

Field Name	**Definition**
Device Identity	Identifier of capture device
Source Address	Origination Address of object
Destination Address	Destination Address of object
Source Port	Origination Port of object
Destination Port	Destination Port of the object
Protocol	Protocol that carried the object
Instance	Canonical count identifying object within a protocol capable of carrying multiple data within a single connection
Content	Content type of the object
Encoding	Encoding used by the protocol carrying object
Size	Size of object
Timestamp	Time that the object was captured
Owner	User requesting the capture of object (rule author)
Configuration	Capture rule directing the capture of object
Signature	Signature of object
Tag Signature	Signature of all preceding tag fields

[0037] For many of the above tag fields in Tables 1 and 2, the definition adequately describes the relational data contained by each field. For the content field, the types of content that the object can be labelled as are numerous. Some example choices for content types (as determined, in one embodiment, by the object classification module 30) are JPEG, GIF, BMP, TIFF, PNG (for objects containing images in these various formats); Skintone (for objects containing images exposing human skin); PDF, MSWord,

Excel, PowerPoint, MSOffice (for objects in these popular application formats); HTML, WebMail, SMTP, FTP (for objects captured in these transmission formats); Telnet, Rlogin, Chat (for communication conducted using these methods); GZIP, ZIP, TAR (for archives or collections of other objects); C++ Source, C Source, FORTRAN Source, Verilog Source (for source or design code authored in these high-level programming languages); C Shell, K Shell, Bash Shell (for shell program scripts); Plaintext (for otherwise unclassified textual objects); Crypto (for objects that have been encrypted or that contain cryptographic elements); Binary Unknown, ASCII Unknown, and Unknown (as catchall categories).

[0038] The signature contained in the Signature and Tag Signature fields can be any digest or hash over the object, or some portion thereof. In one embodiment, a well known hash, such as MD5 or SHA1 can be used. In one embodiment, the signature is a digital cryptographic signature. In one embodiment, a digital cryptographic signature is a hash signature that is signed with the private key of the capture system 22. Only the capture system 22 knows its own private key, thus, the integrity of the stored object can be verified by comparing a hash of the stored object to the signature decrypted with the public key of the capture system 22, the private and public keys being a public key cryptosystem key pair. Thus, if a stored object is modified from when it was originally captured, the modification will cause the comparison to fail.

[0039] Similarly, the signature over the tag stored in the Tag Signature field can also be a digital cryptographic signature. In such an embodiment, the integrity of the tag can also be verified. In one embodiment, verification of the object using the signature, and the tag using the tag signature is performed whenever an object is presented, e.g.,

displayed to a user. In one embodiment, if the object or the tag is found to have been compromised, an alarm is generated to alert the user that the object displayed may not be identical to the object originally captured.

[0040] One embodiment of a process for using the cryptographic signatures is shown in flow chart form in Figure 7. Blocks 702 – 714 describe object capture, while blocks 716 – 722 describe object presentation. First, in block 702, the object is captured by the capture system 22 described above. Then, in block 704, the digital cryptographic signature is generated, e.g., by hashing the object and signing the hash with the private key of the capture system 22. In block 706, all the information gathered by the object assembly module 28 and the object classification module 30 are used to populate the tag fields described above, such as "Source IP," "Content," "Size," and the digital cryptographic signature is inserted into the appropriate "Signature" field.

[0041] In block 708, the cryptographic tag signature over the tag is generated using any known cryptographic method, such as the one used to generate the cryptographic signature over the object in block 704. The cryptographic tag signature is then inserted, in block 710, into the tag in the appropriate "Tag Signature" field. In block 712, the object is stored, e.g., in the content store 44, and in block 714, the tag indexing the object is stored, e.g., in the tag database 42.

[0042] The object and its corresponding tag maintaining the relational data over the object remain stored. Eventually, in block 716, a search – also sometimes referred to as a mine – is performed over one or more of the tag fields that results in a hit on the tag. For example, the tag indicates a particular source IP in the "Source IP" field that was indicated by the search. Next, in block 718, the tag is verified to ensure that it has not

been modified since initial storage in block 714. This can be done by cryptographically verifying the tag signature generated in block 708. For example, the tag signature can be decrypted with the public key of the capture system 22 and compared to a new hash of the tag fields. Other verifications are possible depending on how the tag signature was generated.

[0043] In block 720, the object itself is verified by retrieving the object and cryptographically verifying the digital cryptographic signature stored in the "Signature" field. This can be done similarly to the verification of the tag signature in block 718. In both object and tag signature check out, that is, if neither the object nor the tag has been modified since original storage in blocks 712 and 714 respectively, then the object is presented, in bock 722, to the user. Presenting the object can include any method of delivery, including displaying the object to the user via the user interface 34. If either object or tag has been modified, that is, at least one of the signatures does not check out, the object can still be presented in block 722, but a warning or alert can also be appended to bring the modification to the users attention.

User Interface

[0044] Referring again to Figure 3, in one embodiment, the objects and tags stored in the object store module 32 can be interactively queried by a user via the user interface 34. In one embodiment the user interface can interact with a web server (not shown) to provide the user with Web-based access to the capture system 22. The objects in the content store module 32 can thus be searched for specific textual or graphical content using exact matches, patterns, keywords, and various other advanced attributes.

[0045] For example, the user interface 34 can provide a query-authoring tool (not shown) to enable users to create complex searches of the object store module 32. These search queries can be provided to a data mining engine (not shown) that parses the queries, scans the tag database 42, and retrieves the found object from the content store 44. Then, these objects that matched the specific search criteria in the user-authored query can be counted and displayed to the user by the user interface 34.

[0046] Searches can also be scheduled to occur at specific times or at regular intervals, that is, the user interface 34 can provide access to a scheduler (not shown) that can periodically execute specific queries. Reports containing the results of these searches can be made available to the user at a later time, mailed to the administrator electronically, or used to generate an alarm in the form of an e-mail message, page, syslog or other notification format.

[0047] In several embodiments, the capture system 22 has been described above as a stand-alone device. However, the capture system of the present invention can be implemented on any appliance capable of capturing and analysing data from a network. For example, the capture system 22 described above could be implemented on one or more of the servers 14 or clients 16 shown in Figure 1. The capture system 22 can interface with the network 10 in any number of ways, including wirelessly.

[0048] In one embodiment, the capture system 22 is an appliance constructed using commonly available computing equipment and storage systems capable of supporting the software requirements. In one embodiment, illustrated by Figure 6, the hardware consists of a capture entity 46, a processing complex 48 made up of one or more processors, a memory complex 50 made up of one or more memory elements such

as RAM and ROM, and storage complex 52, such as a set of one or more hard drives or other digital or analog storage means. In another embodiment, the storage complex 52 is external to the capture system 22, as explained above. In one embodiment, the memory complex stored software consisting of an operating system for the capture system device 22, a capture program, and classification program, a database, a filestore, an analysis engine and a graphical user interface.

[0049] Thus, a capture system and content analyser have been described. In the forgoing description, various specific values were given names, such as "packets," and various specific modules, such as the "object assembly module" and "object store module" have been described. However, these names are merely to describe and illustrate various aspects of the present invention, and in no way limit the scope of the present invention. Furthermore, various modules, such as the object store module 32 and the object classification module 30in Figure 3,can be implemented as software or hardware modules, or without dividing their functionalities into modules at all. The present invention is not limited to any modular architecture either in software or in hardware, whether described above or not.